

# Análise de Desempenho de Representações Matriciais em Simulações de Transporte Eletrônico de Grafeno

Alex K. Kodama<sup>1</sup>, Gabriel T. Fonseca<sup>1</sup>,  
Thomas P. Grandin<sup>1</sup>, Calebe P. Bianchini<sup>1</sup>

<sup>1</sup>Faculdade de Computação e Informática (FCI)  
Universidade Presbiteriana Mackenzie – São Paulo, SP — Brasil

{10417942, 10416751, 10418118}@mackenzista.com.br,

calebe.bianchini@mackenzie.br

**Abstract.** *This work evaluates the performance impact of replacing an Intel MKL-based implementation with PETSc in an electronic transport simulation code for twisted bilayer graphene (TBG) in Corbino geometries. The refactoring enabled flexible testing of dense and sparse matrix representations. Initial experiments, using a matrix with approximately 2.92% density, showed that the sparse PETSc version reduced execution time by 79.44% and memory consumption by 30.62% compared with the original code. These results indicate that specialized sparse representations are essential for improving the scalability of TBG nanomaterial simulations.*

**Resumo.** *Este trabalho avalia o impacto da substituição de uma implementação baseada em Intel MKL por PETSc em um código de simulação de transporte eletrônico em grafeno bicamada torcido (TBG) com geometria de Corbino. A refatoração permitiu testar, de forma flexível, representações matriciais densas e esparsas. Nos experimentos iniciais, com uma matriz de densidade aproximada de 2,92%, a versão esparsa em PETSc reduziu o tempo de execução em 79,44% e o consumo de memória em 30,62% em relação ao código original. Os resultados indicam a importância de representações esparsas especializadas para melhorar a escalabilidade dessas simulações.*

## 1. Introdução

Desde a descoberta de estados supercondutores e isolantes correlacionados no grafeno bicamada torcido (TBG, do inglês *Twisted Bilayer Graphene*) em ângulos mágicos, a modelagem física de materiais bidimensionais tornou-se um dos campos mais ativos da física da matéria condensada [Nimbalkar and Kim 2020]. A simulação computacional das propriedades de transporte, como a condutância, nesses sistemas geralmente baseia-se em modelos *tight-binding*, que culminam na montagem e diagonalização de matrizes hamiltonianas de dimensões massivas, especialmente quando aplicadas a geometrias de Corbino [Bahamon et al. 2013]. Consequentemente, o gerenciamento computacional dessas matrizes torna-se o principal gargalo da simulação. A escolha entre utilizar matrizes densas ou matrizes esparsas impacta diretamente a complexidade de memória e o tempo de execução do software.

O código original utilizado para estas simulações dependia da biblioteca Intel Math Kernel Library (MKL), que, embora altamente otimizada para arquiteturas específicas,

apresentava rigidez estrutural quanto aos formatos matriciais aceitos, permitindo apenas o uso do formato *Compressed Sparse Row* (CSR). Para contornar essa limitação e viabilizar um estudo rigoroso das relações entre custo e desempenho computacional, este trabalho propõe a refatoração do sistema para a adoção do PETSc (*Portable, Extensible Toolkit for Scientific Computation*) [et al. 2026]. O PETSc oferece uma abstração de alto nível que permite a troca transparente de formatos de matriz (como blocos densos ou matrizes esparsas comprimidas) por meio de parâmetros de execução.

O objetivo principal deste estudo inicial é avaliar o impacto no desempenho, medindo o tempo de execução e memória máxima alocada pelo programa, ao se alternar entre representações densas e esparsas na simulação da condutância do TBG em geometrias de Corbino, utilizando o ambiente flexível proporcionado pelo PETSc em comparação à implementação original em MKL.

## 2. Metodologia

Esta pesquisa tem caráter aplicado, com abordagem quantitativa e método experimental, visando avaliar o impacto no desempenho de diferentes representações de matrizes.

Inicialmente, foi realizado um estudo bibliográfico sobre álgebra linear computacional, matrizes esparsas e bibliotecas voltadas para computação científica de alto desempenho. Essa etapa teve como objetivo fornecer a fundamentação teórica necessária para compreender as estruturas de armazenamento de matrizes esparsas, os métodos de solução de sistemas lineares e as técnicas de otimização utilizadas em aplicações numéricas.

Em seguida, foi analisado o código desenvolvido por Passos e Bianchini [Passos and Bianchini 2025]. Originalmente implementado em linguagem C, a aplicação emprega a biblioteca Intel Math Kernel Library (MKL) e utiliza exclusivamente matrizes esparsas no formato *Compressed Sparse Row* (CSR). A partir dessa análise, buscou-se compreender a estrutura do programa, o fluxo de execução e as estratégias utilizadas para manipulação de matrizes e resolução dos sistemas lineares.

Após o entendimento do funcionamento do código e dos conceitos relacionados às matrizes esparsas, iniciou-se o desenvolvimento da nova implementação utilizando a biblioteca PETSc. Durante essa etapa, os dados do tipo `MKL_Complex16` foram substituídos pelo tipo padrão C99 `double complex`, além da utilização de objetos do tipo `Mat` fornecidos pela biblioteca PETSc. O solver originalmente baseado no PARDISO foi substituído pelo PCLU (Fatoração LU), em modo de solver direto, mantendo a mesma abordagem empregada no código base. Para operações de álgebra linear básica, foi utilizada a biblioteca OpenBLAS.

## 3. Experimentos e resultados

Apesar de existirem 3 *datasets* para avaliação da simulação, foi utilizado apenas o de menor tamanho devido às limitações do ambiente experimental. O *dataset* é composto por uma matriz de ordem  $1158 \times 1158$  com 39.230 elementos não nulos, representando uma densidade de aproximadamente 2,92%. Os testes foram realizados em um computador pessoal equipado com processador AMD Ryzen 5 5600X e 16 GB de memória RAM, utilizando o ambiente WSL2 Ubuntu versão 22.04.5 LTS, configurado com 7,7 GB de

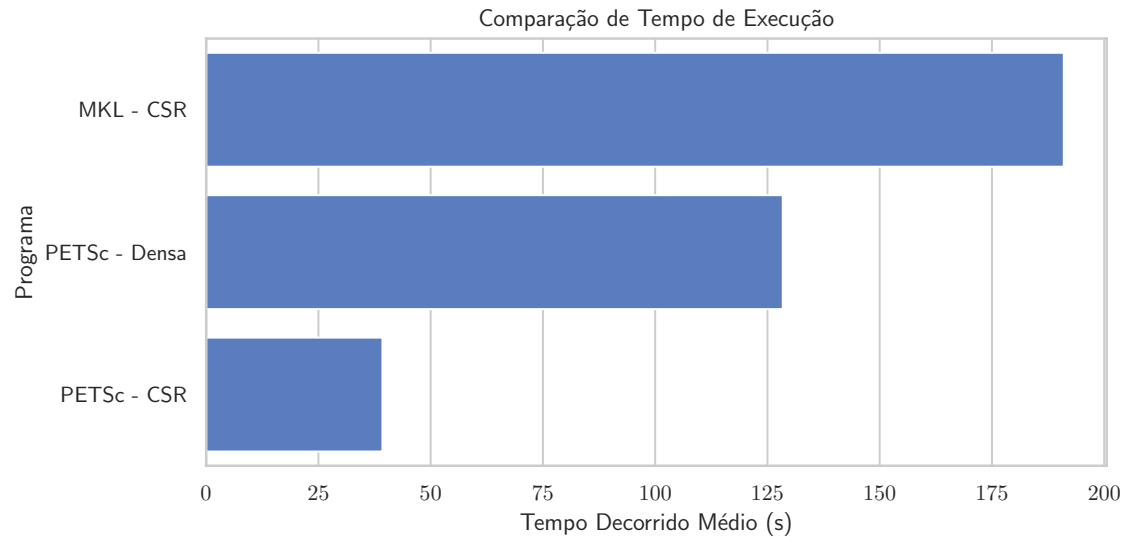
memória RAM alocada. As bibliotecas PETSc, OpenBLAS e Intel MKL utilizadas estavam, respectivamente, nas versões 3.24.6, 0.3.20 e 2026.0.

Ambas as implementações foram compiladas com o compilador gcc versão 11.4.0 utilizando as *flags* de otimização `-O3 -march=native`. Durante os *benchmarks*, os programas foram executados 10 vezes de forma sequencial, sem paralelização. As métricas coletadas para avaliação de desempenho foram o tempo de execução, medido por meio do comando nativo `/usr/bin/time`, e a memória máxima alocada pelo processo (`maxrss`), obtida pelo mesmo comando. A média dos resultados estão dispostas na Tabela 1, junto com o desvio padrão (**DP**) calculado.

**Tabela 1. Resultados de desempenho dos programas**

| Programa      | Tempo (s) | Tempo (s)<br>Desvio Padrão | MaxRSS (KB) | MaxRSS (KB)<br>Desvio Padrão |
|---------------|-----------|----------------------------|-------------|------------------------------|
| MKL - CSR     | 191,00    | 1,74                       | 51.872,80   | 99,09                        |
| PETSc - Densa | 128,41    | 2,12                       | 96.508,40   | 54,24                        |
| PETSc - CSR   | 39,26     | 1,43                       | 35.986,80   | 81,82                        |

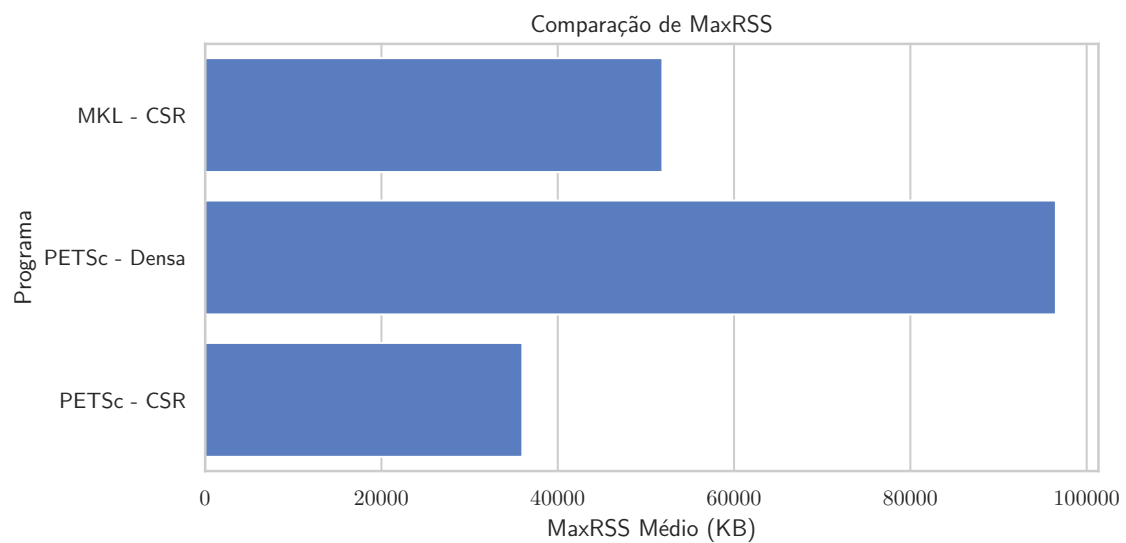
Conforme apresentado na Figura 1, que compara o tempo médio de execução, observa-se uma redução de 32,77% para a versão densa e de 79,44% para a versão esparsa, ambas em relação ao programa original. Similarmente a Figura 2 ilustra o comportamento do `MaxRSS`, evidenciando um aumento de 86,05% no tamanho máximo alocado de memória na implementação densa proposta e uma diminuição de 30,62% na versão esparsa, quando comparadas ao programa original.



**Figura 1. Comparação do tempo de execução.**

#### 4. Conclusão

A substituição da biblioteca Intel MKL pela PETSc permitiu uma maior facilidade para a troca de representações matriciais, mas também demonstrou um ganho computacional em relação ao tempo de execução e memória utilizada, mesmo que mantendo



**Figura 2. Comparação do MaxRSS.**

o tipo CSR utilizado no programa base. Esses resultados mostram a necessidade de se utilizar representações especializadas para a simulação de transporte eletrônico em grafeno bicamada torcido, especialmente em estruturas de Corbino que podem ultrapassar milhões de átomos.

Trabalhos futuros incluem explorar outras representações esparsas disponíveis na biblioteca PETSc, analisar o comportamento da memória cache, assim como executar os demais casos de testes em ambientes mais robustos, como na MackCloud, Laboratório Multidisciplinar de Computação Científica e Nuvem<sup>1</sup>.

## Referências

- Bahamon, D., Castro Neto, A., and Pereira, V. (2013). Effective contact model for geometry-independent conductance calculations in graphene. *Physical Review B*, 88.
- et al., S. B. (2026). PETSc/TAO users manual. Technical Report ANL-21/39 - Revision 3.25, Argonne National Laboratory.
- Nimbalkar, A. and Kim, H. (2020). Opportunities and challenges in twisted bilayer graphene: A review. *Nano-Micro Letters*, 12.
- Passos, A. and Bianchini, C. (2025). Comparação de desempenho de implementações fortran e c: Corbino tbq. In *Anais da XVI Escola Regional de Alto Desempenho de São Paulo*, pages 98–101, Porto Alegre, RS, Brasil. SBC.

---

<sup>1</sup><https://mackcloud.mackenzie.br>